MACM 316 – Computing Assignment 6

Due Date: Friday March 20 at 11:00pm.
Submission Instructions: You must upload one .pdf file in Crowdmark that consists of two pages: page 1 is your report which should fit all discussions, data and figures into a single page; and page 2 is a listing of your code. The deadline is **11:00pm** on the due date. The actual due time is set to 11:05pm and if Crowdmark indicates that you submitted late, you will be assigned a grade of 0 on this assignment. Your TA has emailed you a Crowdmark link that you should save since it will allow you to upload your completed assignments.

- Please review the **Guidelines for Assignments** carefully.
- Acknowledge any collaborations or assistance from colleagues/TAs/instructor.
- If you have any questions about Matlab or aspects of this assignment, then you are strongly encouraged to attend tutorials and drop-in workshops.

Background on Google PageRank (summarized from lectures)

When you do a Google search, the resulting web pages are listed in a ranked order of importance or significance. The method used to compute this ranking is called the PageRank algorithm, and is the basis for Google's financial success. At its core, this algorithm takes the $n \times n$ connectivity matrix G for all web pages on the internet (currently numbered at $n > 60$ billion pages) and computes an eigenvector x that contains the ranks of each page. In this week's computing assignment, you will explore the process of computing the PageRank vector for a few simple web graphs.

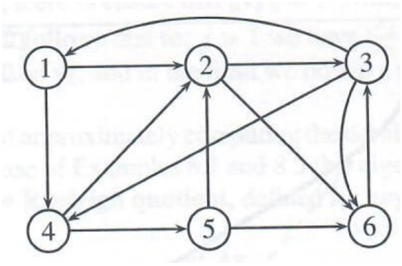
Ranking Web Pages

In lectures you saw a brief description of how the PageRank is computed, which is repeated here. Imagine surfing the web from page to page by randomly choosing an outgoing link from your current page to move to the next. You may occasionally get stuck at a dead end (with no outgoing links) or within a cycle of connected pages. In these cases you might simply "bail out" by picking some other page at random, and then continue from there. Roughly, a page is considered to have high rank if other pages with high rank link to it. Let's expand on what we mean by rank.

The $n \times n$ web connectivity matrix G has entries defined as

$$g_{ij} = \begin{cases} 1, & \text{if there is a link to page } i \text{ from page } j \\ 0, & \text{otherwise} \end{cases}$$

G is enormous but extremely sparse. Suppose that page j points to a total of c_j other pages, then it's clear that $c_j = \sum_{i=1}^{n} g_{ij}$ (the number of *outlinks*) is just the j^{th} column sum of G. Similarly, the row sum $r_i = \sum_{j=1}^{n} g_{ij}$ gives the number of *inlinks* to page i. A small example with $n = 6$ pages is shown below.



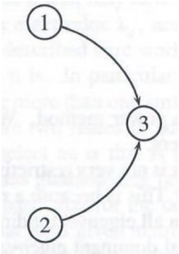
$$G = \begin{array}{c} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \\ c_j = 2 \quad 2 \quad 3 \quad 2 \quad 2 \quad 1 \end{array} \Longrightarrow A = GD = \begin{bmatrix} 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{3} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

Next, define the rank of a page j to be x_j and propose a ranking system in which page j contributes an equal share of its rank, $\frac{x_j}{c_j}$, to each of the pages it points to. We denote for any page i the set of pages that link to it by $\mathcal{B}_i$. Then the rank of this page is just

$$x_i = \sum_{j \in \mathcal{B}_i} \frac{x_j}{c_j}$$

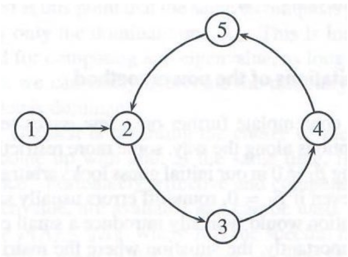
which can be written in matrix form as $x = GDx$ where D is the $n \times n$ diagonal matrix with entries $d_{jj} = \frac{1}{c_j}$. The matrix $A = GD$ with entries $a_{ij} = \frac{g_{ij}}{c_j}$ is shown for the small-web example above. The equation $x = Ax$ has a familiar form, in which our PageRank vector x is an eigenvector of A corresponding to the eigenvalue 1!

This is the simplest situation only, and we still need to explain how to deal with the two problem cases mentioned earlier:



Dead-end nodes: In the example on the right, there is no escaping page 3, which corresponds to a zero column (no outlinks, $c_3 = 0$). A reasonable solution is to simply jump to a random page, which means replacing all entries in the column by $\frac{1}{n}$. Then the matrix A becomes

$$a_{ij} = \begin{cases} \frac{g_{ij}}{c_j}, & \text{if } c_j \neq 0 \\ \frac{1}{n}, & \text{if } c_j = 0 \end{cases}$$



Cyclic paths: For the example on the right, one can get trapped in the cycle $2 \to 3 \to 4 \to 5 \to 2 \to \dots$. To permit escape, we turn our systematic web traversal into a *random walk* in which we occasionally skip the usual page selection and instead jump to some arbitrary page in the web. To be more precise, we follow links as usual with some probability p (say $p = 0.85$), but then with probability $1 - p$ we jump to another randomly selected page (p is sometimes called a "teleport probability"). This leads to a modified formula for A:

$$a_{ij} = \begin{cases} \frac{p g_{ij}}{c_j} + \frac{1-p}{n}, & \text{if } c_j \neq 0 \\ \frac{1}{n}, & \text{if } c_j = 0 \end{cases}$$

In summary, finding the PageRank vector x reduces to solving the eigenvector problem $x = Ax$ where A can be written compactly in matrix form as

$$A = pGD + ez^T, \qquad \text{with } z_j = \begin{cases} \frac{1-p}{n}, & \text{if } c_j \neq 0 \\ \frac{1}{n}, & \text{if } c_j = 0 \end{cases}$$

and $e = (1, 1, \ldots, 1)^T$ being the n-vector containing all ones.

Computing the PageRank Vector with Matlab

A simple approach for solving eigenvalue problems is called the *power method* and takes the form of a fixed point iteration $x^{(k)} = Ax^{(k-1)}$ for $k = 1, 2, \ldots,$ given some initial guess $x^{(0)}$. With no prior knowledge about the ranks, we choose $x^{(0)} = \frac{1}{n}e$ which assumes that all pages are equally ranked. And that's it![1] The algorithm described above is fairly easy to implement in MATLAB, using the code below (posted on Canvas as `mypagerank.m`):

```
% Estimate the PageRank for a small web connectivity
% matrix using the power method.

p = 0.85;  % "teleport" probability

% Set up the connectivity matrix using index
% vectors that define the from->to connections
% between pages numbers 1 to n.
n = 6;     % number of pages
ii = [2 4 3 6 1 4 6 2 5 2 6 3];  % "to" index
jj = [1 1 2 2 3 3 3 4 4 5 5 6];  % "from" index
G = sparse(ii, jj, 1, n, n);

c = full(sum(G));  % column sums
e = ones(n,1);
k = find(c˜=0);    % indices of zero columns
D = sparse(k, k, 1./c(k), n, n);
z = ((1-p)*(c˜=0) + (c==0)) / n;

A = p*G*D + e*z;   % PageRank matrix

x = e/n;           % initial guess
xold = zeros(n,1);
niter = 0;
while norm(x-xold) > 0.0001,
  niter = niter + 1;
  xold = x;
  x = A*x;
end
x, niter
bar(x), shg
```

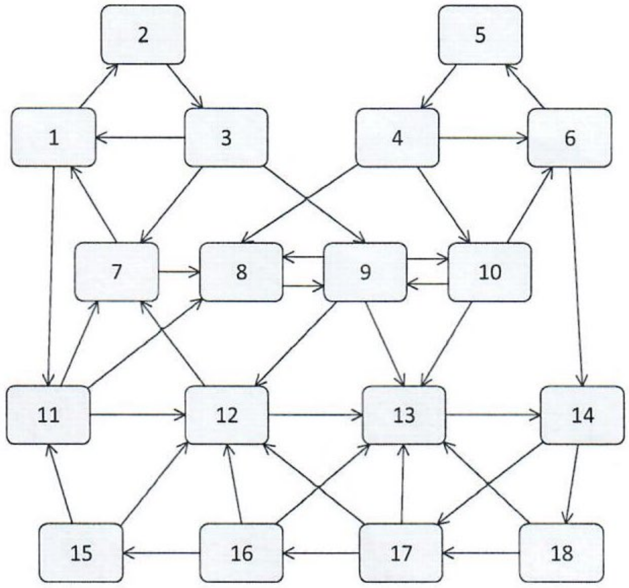
This code returns the PageRank vector $x = (0.1038, 0.1693, \underline{0.2781}, 0.1479, 0.0879, 0.2131)^T$, from which it's easy to see that page 3 receives the top ranking.

[1] This is a simplified version of the power method that applies to the very special case where A has positive entries, $\|A\|_1 = 1$, $\|x^{(0)}\|_1 = 1$ and the eigenvalue is 1. Under these conditions, we're guaranteed to have a unique positive real solution x, and the power method converges to this solution, albeit sometimes slowly. For more general eigenvalue problems, the power method requires requires a few extra modifications.

Your Computing Assignment – Exploring PageRank

1. Run the `mypagerank` code on the given 6-page example for various values of the teleport parameter p lying between 0 and 1. How do changes in p impact the top-ranked web page, and the speed of convergence of the power method iterations? What happens to the rankings as $p \to 0$, and can you explain why this might be so?

2. Modify the code for the larger set of $n = 18$ web pages and links pictured below:



   (a) Compute the PageRank for each of the 18 pages using $p = 0.85$, and report the results of the ranking as a bar plot (you may find MATLAB's `sort` command useful for picking out the highest-ranked page).

   (b) Repeat with $p = 0.5$. Which page ranks increase, and which decrease?

   (c) Repeat with $p = 0.98$, and notice that several pages have a rank that is close to zero. Is there anything about the link structure of this web graph that might explain why this happens? If you were able to add a single link between any two web pages that would most increase the reputation of the lowest-ranked pages, which would it be? Explain your choice.